Integrating multiple modalities into SLMs and parsing the output of SLMs

Karl Weilhammer, Rebecca Jonson, Håkan Burden,
Jost Schatzmann, Matt Stuttle and Steve Young

Distribution: Public

TALK
Talk and Look: Tools for Ambient Linguistic Knowledge
IST-507802 Deliverable 1.4

August 14, 2006

Project funded by the European Community
under the Sixth Framework Programme for
Research and Technological Development

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	IST-507802
Project acronym	TALK
Project full title	Talk and Look: Tools for Ambient Linguistic Knowledge
Instrument	STREP
Thematic Priority	Information Society Technologies
Start date / duration	01 January 2004 / 36 Months

Security	Public
Contractual date of delivery	M30 = June 2006
Actual date of delivery	August 14, 2006
Deliverable number	1.4
Deliverable title	Integrating multiple modalities into SLMs and parsing the output of SLMs
Type	Report
Status & version	Final 1.0
Number of pages	33 (excluding front matter)
Contributing WP	1
WP/Task responsible	UCAM
Other contributors	UGOT
Author(s)	Karl Weilhammer, Rebecca Jonson, Håkan Burden, Jost Schatzmann, Matt Stuttle and Steve Young
EC Project Officer	Evangelia Markidou
Keywords	Grammar, Language Model, Speech Recognition

The partners in TALK are:

Saarland University	USAAR
University of Edinburgh HCRC	UEDIN
University of Gothenburg	UGOT
University of Cambridge	UCAM
University of Seville	USE
Deutches Forschungszentrum fur Künstliche Intelligenz	DFKI
Linguamatics	LING
BMW Forschung und Technik GmbH	BMW
Robert Bosch GmbH	BOSCH

For copies of reports, updates on project activities and other TALK-related information, contact:

The TALK Project Co-ordinator
Prof. Manfred Pinkal
Computerlinguistik
Fachrichtung 4.7 Allgemeine Linguistik
Postfach 15 11 50
66041 Saarbrücken, Germany
pinkal@coli.uni-sb.de
Phone +49 (681) 302-4343 - Fax +49 (681) 302-4351

Copies of reports and other material can also be accessed via the project's administration homepage, `http://www.talk-project.org`

Contents

Summary

The original focus of task 1.4 was to develop techniques by which multimodal input streams can be integrated into SLMs. Analysis of a Wizard of Oz data collection showed that integrated multimodal interaction of users with the dialogues system occured only very rarely. In these rare cases, when speech and clicks were used in an integrated way, clicks mostly occured seconds before or after the speech, there were only few cases where both overlapped. This leads to the conclusion, that the statistical language model (SLM) of a speech recogniser is not the right place for integrating multimodal inputs. It should be done at a later stage. The results of task 1.3 suggest that using SLMs instead of grammars leads to superior speech recognition performance. If SLMs are used in a dialog system along with late integration of multimodal acts, robust semantic parsing of speech recognition output is necessary. With the agreement of the reviewers, the scope of WP1.4 was widened to include robust semantic decoding. In this deliverable, we explore methods to create statistical semantic parsers or dialogue act taggers by generating corpora from application grammars using the Grammatical Framework. We investigate approaches based on memory based learning, an n-gram based semantic parser and the Hidden Vector State Model. We create all statistical models directly from our interpretation grammars and compare f-measures or accuracies of semantic concepts or dialogue acts. The results show an important improvement of statistical methods over classical grammars. Our experiments were carried out in English on a tourist information task and in Swedish on a MP3 player interface. For both sites involved in this deliverable the results of this deliverable have great influence on the decision of which robust semantic parser they will include in their final system.

Chapter 1

Introduction

This document reports mainly about parsing the output of Statistical Language Models (SLMs). The original focus of task 1.4 was to develop techniques by which multimodal input streams can be integrated into SLMs. As already detailed in status report T1.4s2 [YSW05], analysis of the SACTI multimodal Wizard Of Oz data collections lead to the following results:

- Less than 2.5% of the data collected feature integrated multimodal dialog acts.
- Only 8 of the 36 participants used these acts.
- The time relation between speech and clicks is not on the word level, but rather on the utterance level.

This suggests integration of multimodal acts should preferably not be done in the language model component of the recogniser, but at a later stage. A grammar based recogniser has the advantage that semantic decoding can already be done inside the recognition grammar. The results of T1.3 [WJRY06] suggest that using SLMs instead of grammars leads to superior speech recognition performance. However, if SLMs are to be used in a dialog system, and late integration of multimodal acts is preferred, there exists the issue of parsing grammatically less restricted speech recognition output. Robust semantic parsing remains an active research problem even for unimodal input and forms a barrier for further progress with the current systems in Cambridge and Gothenburg. With the agreement of the reviewers, the scope of WP1.4 was widened to include semantic decoding, following the outcome of task 1.3. As status report T1.4s2 [YSW05] already contains a detailed discussion on the problem of integrating multimodal input streams in SLMs, this document will concentrate on semantic decoding.

1.1 Semantic Modelling of SLM-based ASR output

Taking the results of deliverable D1.3 [WJRY06] seriously means that speech recognisers should use statistical language models instead of recognition grammars. In such a system architecture it is necessary to introduce a new component, a semantic decoder, that can convert user speech (orthographic form) into dialogue acts the system can process. With a grammar this is usually a trivial operation or a function of the grammar itself. When using output from a SLM-based ASR system, the problem of parsing the input into a dialogue act presents a significant challenge.

The Cambridge group researches statistical parsers because they would fit best into the architecture of a fully statistical dialogue system. The advantage of an entirely statistical approach is that dialogue acts and probabilities are passed to the dialogue manager and the dialogue manager can take these probabilities into account when deciding on which action to choose next.

The Gothenburg group is investigating statistical parsers for the following reason: If a statistical language model is used instead of a recognition grammar, then a lot of unexpected expressions will be in the speech recognition output. Grammatically unconstrained strings are not well suited to be parsed by a grammar. This means that using robust statistical parsers instead of a classical GF parser for semantic decoding is a good alternative.

Figure 1.1: *Speech understanding component consisting of an ATK speech recogniser using SLMs and a Hidden Vector State semantic parser (HVS).*

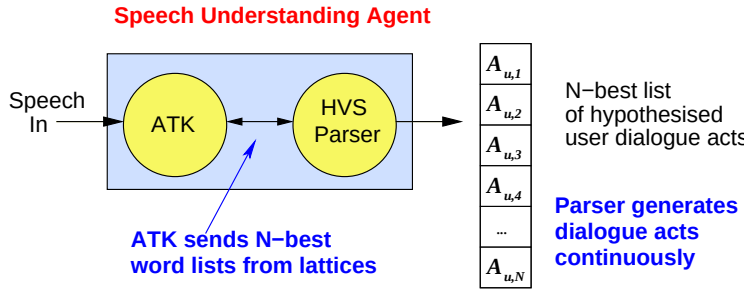


Figure 1.1 shows an example of how a semantic parser is integrated with a speech recogniser to form a speech understanding component. Here the ATK recogniser [You04] passes n-best lists to the semantic parser realised as a Hidden Vector State Model. The speech understanding component then passes a list of dialogue acts to the dialogue manager.

1.2 Existing Rule Based and Statistical Approaches

Existing approaches for semantic parsing can be grouped in a continuum between rule based, deterministic models and statistical models.

Rule-based systems typically require hand-crafted rules which are integrated in grammars that are designed to parse full sentences or chunks of sentences. The most prominent of these are MIT's TINA [Sen92], CMU's PHOENIX [WI96], SRI's Gemini [DMAM94] systems. Sometimes these rule based systems are then augmented with corpus statistics to get better results. Whilst good performance is often achieved using this approach, rule-based parsers are normally expensive to build and hard to transplant from one application to another. Furthermore, they can still degrade badly in the face of high speech recognition error rates and unexpected or ill-formed input sentences. In TALK we have used the Grammatical Framework (GF) [Ran04] both to produce recognition grammars and SLMs as well as for semantic parsing. However, at the current time there are no possibilities for robust parsing within GF which would be needed for an optimal performance when using SLMs for recognition. To implement robust parsing in GF is beyond the scope of the TALK project and constitutes a research area of its own. However, in Chapter 4 we propose different possibilities for achieving robust parsing using GF grammars in the future.

The statistical approach seeks to automatically train parsers from semantically annotated sentences in the hope of building more robust decoders with less effort. An early example is AT&T's finite state semantic tagger in which a HMM is used to assign semantic concepts to words [PTG+92]. More sophisticated models have been proposed since that can handle hierarchical structure such as the hierarchical Hidden Understanding Model [SMSM97] and the hierarchical Hidden Markov Model [FST98]. These models typically require every training utterance to be semantically annotated. More recently the Hidden Vector State model has been proposed as a constrained hierarchical model which does not require detailed semantic annotation to train [HY05]

An alternative approach treats semantic decoding as a straightforward pattern recognition problem. The Y-Clustering parser [YY06] classifies each input sentence to a set of exemplar sentences, which were automatically selected from a training corpus using a sentence clustering technique.

In this report one of the existing techniques, the HVS model, is compared with a new model based on a tagged statistical language model. The particular focus of this study is the effectiveness of the models when bootstrapped from an automatically generated corpus.

Dialogue act (move) tagging has been of great interest mostly in order to be able to annotate corpora with dialogue acts automatically [SCVS98] but also in some cases to decode or even predict the dialogue move of the user's last utterance in dialogue systems [SCB+00]. Dialogue move tagging has been explored with different statistical and machine learning techniques [SCVS98, SCB+00, LdBKC04]. The best dialogue act tagging models have obtained an accuracy of 70% showing the difficulty of the task of classifying utterances to dialogue acts [PM98]. As different studies use different types and different amount of dialogue acts it is hard to draw any comparative conclusions from previous work. A survey of related work shows that the predictors used for dialogue move tagging also vary. In [PB96] the prediction is dependent on the system's last dialogue act and the range of user acts corresponding to each system act seems to be chosen manually beforehand. In the Verbmobil project [REKK96] the next dialogue act was predicted by using statistical language models holding a dialogue history of previous dialogue acts. They also included directional information by including the speaker as a tag. They used annotated logs for training and were able to classify 18 distinct dialogue acts with an accuracy of 40%. In three subsequent studies shown in [WPI99, PKIW98, PM98] dialogue move prediction was done with statistical language models trained on annotated dialogues (the MapTask Corpus) experimenting with different predictors such as e.g. intonation features of the user utterance or game information apart from dialogue move history and speaker information. In [PM98] twelve distinct moves were classified with an accuracy of 57% by using handcoded game information.

In this report we will use data generated from grammars which means that we need not put any effort on manual tagging to obtain annotated data and at the same time we can assure that our data includes all possible dialogue moves that we want to be able to decode in our domain. However, we will not be able to use any additional features. Another distinction of our work is that we are not tagging abstract dialogue moves such as answers, requests etc. as in previous work but we are tagging combination of dialogue moves and their slots i.e. dialogue moves such as ”answer(group(abba))” or ”request(playlist_add), answer(song(dancing,queen))”. This is a much harder task.

1.3 Report organisation

The main focus of our experiments was to investigate if statistical semantic parsers can be trained with corpora generated by GF task grammars. The experiments were carried out in two different tasks, a MP3

player interface and a tourist information system. The task languages were Swedish and English. Chapter 2 details results on dialogue act tagging using memory based learning. These experiments were carried out in Swedish on a MP3 player domain. The used dialogue act taggers provide only a shallow annotation. They can not capture deep semantic hierarchies. Chapter 3 contains work on semantic parsing for an English dialogue system designed for a tourist information task. The semantic models that are investigated are the Hidden Vector State model and a novel n-gram based parser. Both models are hierarchical semantic parsers using a theoretical model that is capable of discovering full semantic trees. Both models were evaluated against deep (full tree) and shallow (leave nodes) criteria. Finally, Chapter 4 shortly summarises the results of both investigations.

Chapter 2

Bootstrapping a Dialogue Move Tagger

In D1.3 [WJRY06] we showed how to bootstrap statistical language models (SLMs) by generating training corpora from GF grammars. This resulted in an enhanced speech recognition performance for the DJ-GoDiS application as shown in [Jon06]. However, using unconstrained SLMs instead of restricted recognition grammars means that the output from the speech recogniser will also be unconstrained and thus the GF grammar will not always be able to parse the ASR output. Our preliminary strategy for handling this problem in the DJ-GoDiS domain was to use a simple rule-based parser written in Prolog that looks for keywords and phrases and maps them to dialogue moves (just as was done before the integration of GF and GoDiS, see [Lar02]). However, such a parser is hard to maintain and doubles the grammar work. In this chapter, we will therefore show how we have bootstrapped a dialogue move tagger using the same methodology as in D1.3. We have trained two different dialogue move taggers from a corpus generated from GF using memory based machine learning.

2.1 Training and Test Data

The main difference with our tagger approach in comparison with previous work is that we have trained our taggers on a corpus generated from a GF grammar [Ran04] written for the domain where all utterances appear together with the dialogue move(s) they should be interpreted as. In this experiment we have focused on the GoDiS application DJ-GoDiS which is a multimodal MP3 player that lets users control their Internet audio player with the voice or by graphical input [EAB+06]. The user can among other things change settings, choose stations or songs to play or create playlists. The data used for the experiment comes from corpora generated automatically by GF from the Swedish GF grammar written for the dialogue system DJ-GoDiS reported in [LBC+05].

The number of abstract dialogue moves in GoDiS is limited to requests, answers, ask moves, greetings, quit moves and moves involving Interactive Communication Management (ICMs), e.g. grounding moves, as reported in [Lar02] and in [LBC+05]. In DJ-GoDiS we have 6 different types of answers (e.g. `answer(song(X))` where X can be any of the songs in the MP3player), 3 different types of ask moves and 18 different requests. In GoDiS an utterance and a dialogue move is not necessarily a one-to-one pair but an utterance can be interpreted as several dialogue moves i.e. conveying several concepts. This means that the number of possible dialogue move combinations gets very large. The GF grammar that we have used distinguishes 3873 dialogue move combinations and holds 55702 utterances representing these

which implies that we have 55702 training instances marked with dialogue moves in our training corpus. In this study we have focused on the Swedish grammar and generated Swedish utterances. However, the English grammar would have given the same dialogue move combinations as the grammars have a common abstract level. A corpus fragment generated with GF from an early version of the English grammar follows below to show the format of the original training data where all dialogue moves were generated together with the utterances the grammar covered for these moves.

```
[request( playlist_add ), answer(item([the,final,countdown])),
answer(group([europe]))]
i want to add the final countdown with europe please
i would like to add the final countdown with europe please
i want to add the final countdown with europe
i would like to add the final countdown with europe
add the final countdown with europe please
add the final countdown with europe
i want to add europe with the final countdown please
i would like to add europe with the final countdown please
i want to add europe with the final countdown
i would like to add europe with the final countdown
add europe with the final countdown please
add europe with the final countdown
```

Our taggers have been tested on a test set of 263 transcribed and annotated Swedish user utterances including both unknown words and unknown constructions. These user utterances were collected with the DJ-GoDiS system and thus represent the type of input a semantic decoder for this domain could be exposed to. The utterances vary in length and range from simple one-word utterances (e.g. yes answers) to more complicated twelve word utterances. An excerpt from the test set with dialogue move tags is found below. The test set was tagged manually with dialogue moves by two annotators with an inter-annotator agreement of kappa 0.99 [Car96].

```
jag vill fråga om vilka låtar han har gjort
(Eng. i want to ask about what songs he has made)
ask(x^songs_by_artist(x))

lägg till sommaren är kort med tomas ledin
(Eng. add sommaren är kort with tomas ledin)
request(playlist_add) + answer(item([sommaren,är,kort])) +
answer(group([tomas,ledin]))

sommaren är kort
(A song title)
answer(item([sommaren,är,kort]
```

2.2 Dialogue Move Tagging

We have built two different taggers to simulate a more robust way of parsing. Both were trained on the corpus generated from GF where all utterances appear together with the dialogue move(s) they should be interpreted as. The first tagger is utterance-based and built with the memory-based learner TiMBL [DZvdSvdB01]. Although, this seemed to work successfully we opted for building a second tagger with MBT [DZvdBvdS03], as it gave us a tagger we could use directly at run-time and that would make it able to give us dialogue move scores as explained below.

2.2.1 Utterance-based Dialogue Move Classifier

This tagger or dialogue move classifier was trained on 55702 utterances represented as bag of words (BoW) and additionally the length of the utterance which in total gives a vector of 237 features. The BoW is as big as the corpus vocabulary and holds a position for each word which will be marked as 1 when the word appears in the utterance. A feature vector example representing a request to play a specific song (vara vänner) by a certain artist (jakob hellman) follows below where the first number means that the utterance consists of ten words, and that the words correspond to the positions in the BoW marked with 1 and should be interpreted as the dialogue move tags `request(start_specific)`, `answer(item([vara vänner]))`, `answer(group([jakob hellman]))`:

```
10,[börja,toppnivå,glömma,man,kan,ha,hjälp,få,avbryta,musiken,
stäng,stopp,stoppa,visa,bakåt,igen,spelningen,återuppta,allt,
listan,rensa,höj,höja,viss,speciell,start,början,från,paus,
pausa,ljudet,sänk,volymen,sänka,radiostation,välja,spelaren,
prata,framåt,spola,avsluta,sluta,hejdå,hörde,förlåt,va,sa,
ursäkta,jaha,visst,ok,okej,inte,hallå,tjena,hej,nu,spelas,
heter,japp,jajamen,ja,nepp,nä,nej,ettan,höger,vänster,skifta,
mitten,balansen,ändra,bort,ta,tredje,tionde,sjätte,sjunde,
andra,nionde,fjärde,första,femte,åttonde,föregående,nästa,
den,tre,tio,sex,sju,låt,nio,fyra,fem,nummer,lyssna,höra,1,
spela,radio,rant,stationen,gunfire,digital,lägg,spellistan,1,1,1,
till,lägga,låten,skrivit,de,gjort,han,fråga,någonting,låtar,
vilka,artisten,har,vad,ytan,under,moln,ett,segla,vingar,grader,
hundra,åtta,tro,ska,vindarna,diamanter,vill,göra,får,vet,vem,
här,var,två,tv,på,flickorna,tunga,kärlekens,kråkan,och,flickan,
hörnet,runt,himlen,du,som,precis,om,håll,landskap,öppna,finns,
det,vargar,jagad,hellre,blir,1,mig,ihåg,kom,rummet,i,ängeln,
sarah,kort,är,sommaren,1,1,hjärta,mitt,av,del,en,solglasögon,
lundell,ulf,leva,di,svenningsson,uno,lemarc,peter,jackson,
michael,nilsson,rickfors,madonna,wiehe,mikael,ryde,annelie,
lakejer,lustans,grön,ebba,1,1,ledin,tomas,tider,gyllene,imperiet,
freda,dahlgren,eva,eldkvarn,orup,kent,irma,död,docent,isaksson,
patrik,ekdahl,lisa,winnerbäck,lars,orkester,kaspers,bo,1,undantag],
[request(start_specific),answer(item([vara,vänner])),
answer(group([jakob,hellman]))].
```

We tested the tagger against our test set of manually tagged user utterances from real DJ-GoDiS interactions. The tagger showed a 79% accuracy on the test set where 156 were exact matches (i.e. existed in the training corpus and likewise in our original GF grammar). These exact matches could be seen as the grammar coverage giving a 59% accuracy which means that we have been able to boost the performance getting a more robust interpreter by using the grammar corpus as training data. This means that we get 34% increase in tagging accuracy by using the bootstrapped tagger (significant at $p < .0001$) instead of the GF grammar. We get a more robust behaviour than with the tagger and is able to interpret unexpected expressions that are similar to the training data. A closer look at the tagging results shows that the tagger even manages to give a partly interpretation to utterances including unknown songs i.e. an utterance such as ”I want to add UNKNOWN” will get the tag `request(playlist_add)`. This means that the dialogue manager will be able to take the dialogue a step forward. This would not be possible with the grammar which would fail in giving any semantic interpretation of the utterance at all.

However, this tagger does not take into account word order which means that ”John saw Mary” will be tagged the same way as ”Mary saw John”. In our domain this order does not really matter for cases like ”Abba with Dancing Queen” or ”Dancing Queen with Abba” (both interpreted as `answer(group(abba))`,`answer(song(dancing queen))`) as long as we do not have artists or songs with the same name. However, in many other domains, of course, we need to be able to make this distinction. Simple cases can be solved by having an additional Bag of Bigrams (BoBi), where the bigram ”John saw” would have a position and would be marked in the first case but not in the second case where ”Mary saw” would be marked instead. For the moment the utterances in this domain are simple enough to do without this extension but a more advanced technique would be needed if you want to do more advanced parsing.

A TiMBL classifier does not only give a class (in this case a dialogue move or dialogue move combination) as output but can also give a confidence score for its choice. Our classifier could therefore be used to tag utterances together with a confidence score given from TiMBL for the choice of dialogue move tag. In this way we could just reject dialogue moves with a confidence score which is too low and in that way avoid some of the incorrect tags. Additional training data could be obtained from dialogue system logs where DJ-GoDiS were run with the simple Prolog parser. Using this material could improve the accuracy even further. However, in this case we used the existing logs as test data.

2.2.2 Word-based Dialogue Move Tagger

The second tagger was generated with MBT (memory-based-tagger) [DZvdBvdS03]. The tagger generator MBT is normally used to develop part-of-speech (POS) taggers. We have used it to be able to decide what dialogue move a word in an utterance belongs to. As training data we used the GF corpus converted into a format where each line holds a word and a dialogue move. The utterance “lägg till abba på spellistan” (Eng. add Abba to the playlist) is represented as follows:

```
<utt>
lägg request(playlist_add)
till request(playlist_add)
abba answer(group(abba))
på request(playlist_add)
spellistan request(playlist_add)
</utt>
```

We generated a tagger that for known words takes into account two tags before the focus word to be tagged and two words after. For unknown words the tagger looks at the previous tag and at the first four letters of the focus word for clues. This means that the tagger can tag unknown words correctly by identifying a known lemma (e.g. “lägga” (Eng. Add) if “lägg” (Eng. Add) is known). This contextual feature set was chosen in a development phase. Enlarging the context on either side when tagging known words did not give any improvement but two words back and two words ahead seemed to be optimal. For unknown words we also tested looking at suffixes but although suffixes normally are useful for the task of POS-tagging it did not seem to be very useful for dialogue move tagging of Swedish words where the content part of the words are more important and is usually not placed in the end of the word. At runtime the tagger can be fed with utterances followed by the sentence delimiter `<utt>`. The output of the Swedish phrase “jag vill lägga till orup” (Eng. “I want to add Orup”) looks as follows:

```
jag/request(playlist_add)
vill/request(playlist_add)
lägga/request(playlist_add)
till/request(playlist_add)
orup/answer(group([orup]))
```

As seen, each word will get a dialogue move tag. Unknown words will also get a tag but will be indicated with // instead of /. The tagger has been tested on the manually dialogue move tagged test set of transcribed user utterances which included for the GF grammar both unknown words and unknown constructions. The tagger has a 79% tagging accuracy (84% for known words) on this test set of 263 utterances where 156 are exact matches (i.e. existed in the training corpus). These exact matches could again be seen as the grammar coverage which gives a 59% accuracy which means that we once again have been able to boost the performance, getting a more robust interpreter with a 34% increase in tagging accuracy. As seen, this tagger performs similarly to the previous tagger.

The word based tagger seems to have a problem when common words occurring in song titles appear alone (such as you, a etc.) tagging them rather as belonging to a song title instead of the overall dialogue move. It seems that it has been over-trained on songs and groups. This could be solved by a post-process checking if the rest of the song title words really appear in the utterance. Another option is to retrain the tagger with songs and groups represented as whole entities (e.g. dancing_queen).

2.2.3 Dialogue Move Scores

The word based tagger makes it possible to calculate what we call “dialogue move scores” by taking the word confidence scores from the ASR for all words tagged with a specific dialogue move and calculate the mean confidence of these. This means that for the example above we would get two scores: one for the dialogue move `request(playlist_add)` based on the word confidences of four words and one for the move `answer(group(orup))` based on the word confidence of the artist “orup”. In this way we will not rely on the ASR confidence score for the whole utterance when choosing grounding strategies in GoDiS but look at the confidence score

for each dialogue move. The GoDiS system uses a more fine-grained scale of grounding levels than many other dialogue systems and the grounding behaviour in GoDiS is not limited to the perception level but also chooses different strategies dependent on semantic and pragmatic understanding of the user input (see [Lar02]). In GoDiS each dialogue move is grounded separately and the choice of grounding strategy is currently conditioned on the confidence score from the speech recogniser for the whole utterance. However, it is often the case that some parts of an utterance have a higher confidence rating than others (this is shown by the variation of the word confidence scores) and a better dialogue behaviour would be for example to confirm only the parts rated lower. This is easily done if we can obtain dialogue move scores. The GoDiS grounding behaviour would work in the same way only with the minor modification that we keep track of each dialogue move's score instead of only the ASR confidence score. A multi-score version of GoDiS has therefore been implemented to be prepared for the use of dialogue move scores. From a multimodal point of view this means that we can assign optimal confidence to dialogue moves performed through the graphical input such as a click. This makes it possible to avoid explicit grounding of these moves.

2.2.4 Tagging N-Best Lists with Dialogue Moves

To be able to robustly tag N-Best hypotheses and their transcriptions with dialogue moves for a re-ranking experiment we used the word based dialogue move tagger to simulate a more robust way of parsing as we are using the SLM generated in D1.3 to produce the N-Best lists.

We will not go into detail of the re-ranking experiment as it is beyond the scope of this deliverable but we will give a short description to see how the tagger was used. This re-ranking experiment is a further elaboration and adaptation to the GoDiS environment of the work carried out in [MAB+05] (see chapter 6). It shows how we can benefit from taking into account dialogue context when re-ranking speech recognition (ASR) hypotheses. We have carried out experiments with human subjects to investigate their ability to rank ASR hypotheses from the DJ-GoDiS domain using dialogue context. Based on the results of these experiments we have explored how an automatic machine-learnt ranker profits from using dialogue context features. An evaluation of the ranking task shows that both the human subjects and the automatic classifier outperform the GoDiS baseline (i.e. always choosing the topmost of an N-Best list) and that they perform better and better the more dialogue context is made available. Actually, the automatic classifier performs slightly better than the human subjects and reduces sentence error rate 53% in comparison to the baseline.

What we needed for the experiment to prepare the training data was a robust way to tag the N-Best hypotheses and the manual transcriptions of the user utterances. We used the grammar to get a grammaticality score but opted for the word based tagger for the tagging task. The test set used in the above experiments is more extreme than the N-Best lists we have used here as the words in the N-Best lists are all known to the tagger as the vocabulary for the SLM is the same as for the original GF grammar. We used the word based tagger to tag the 2654 ASR hypotheses in our training data and the 391 transcriptions with dialogue move tags for each word. We then took the word dialogue move tags and eliminated all duplicates to get a dialogue move tag (or

tags) for the whole utterance. This was used as one of the features for our machine classifier. Another feature that we were able to obtain was a list of dialogue move scores calculated from the dialogue move word tags and the word confidence scores as explained earlier. We also looked at the resulting dialogue moves in an N-Best list and picked out the most frequent dialogue move of the list as an additional feature. All these features obtained with help of the dialogue move tagger resulted in being very important factors for the re-ranking task.

We also used the dialogue move tags to be able to compare each hypothesis with the transcription on concept level and by that automatically label all hypotheses as being conceptually similar or not to the transcription.

2.3 Dialogue Move Prediction

In D1.3 we generated dialogue move specific language models showing that by using these we would be able to improve recognition performance even further. However, to be able to use these we need to be able to choose between them i.e. we need to predict what dialogue moves are to come.

Out of the scope of task 1.4 we have carried out some work to explore dialogue move prediction i.e. predicting what the user of a dialogue system may do in his/her next turn. We have used the machine learner TiMBL [DZvdSvdB01] to predict user dialogue moves from information states. Our first experiment was based on a small training data set of automatically generated logs of information states and dialogue flow from the DJ-GoDiS application. The features considered for the experiment were chosen from the information available in the information state in the dialogue logs. The features selected were the previous move (PM) (i.e. the move before the current system move), the information in shared commitments (SHCOM), the shared actions (SHACT), the current question under discussion (QUD) and the current system move (LM).

An example of a training instance where the dialogue state is represented by the five features explained above and classified with the next user dialogue move looks as follows:

```
ReqList,=,Add,WhGroup,ICM@ICM@AskArtist,AnsGr@AnsSo.
```

This corresponds to a dialogue state where:

- the previous move was a request concerning the playlist (ReqList)
- there are no shared commitments (=)
- there is a shared action of adding something to the playlist (Add)
- the question under discussion is what group to add
- the current move is a combined move of grounding moves (ICM) and a question about what artist is under consideration (AskArtist).

- The user move performed in this case was a combination of two answers: the name of a group and of a song.

We obtained an accuracy of 67.51% by using the information state and classifying 19 different classes. The classes here correspond to different dialogue move combinations that could be of interest to gather in the same dialogue move specific SLM (DMSLM). This means that we could distinguish between 19 different types of DMSLMs.

Another thing to keep in mind when looking at the results is that there are no uniquely correct matches of a dialogue move and a state as a user can choose between several possible moves in each state. What we want is a learner that can predict the most plausible move (or moves) to help us choose an appropriate language model. To get a better idea of how our learner is working we need to look at its top choices and see if one of these corresponds to the user move which was actually realised. Using the TiMBL verbose option "db" gave us results where this could be investigated and from which we could calculate a more appropriate accuracy score for the task. By only looking at the 1-Best result we got an accuracy of 67% percent. However, by considering the two best choices the learner gives us we get an accuracy of 75%. Looking at the learner's three best choices to see if the correct class is among these gives us an accuracy of 81%.

The work described here for predicting the dialogue moves a user may perform in order to be able to switch to appropriate language models could easily be adapted to the similar task of decoding the dialogue move performed by a user. In this case we could use the same information as has been used here but in addition we would have information from the speech recogniser available. This could mean that that our decoding task gets much easier with more information available than just a text string as in section 2.2.

2.4 Conclusions

We have shown that we can bootstrap dialogue move taggers in the same way we bootstrapped statistical language models in D1.3 by generating training corpora from GF grammars. Our dialogue move tagger performs better than our interpretation grammar just as the SLMs performed better than the ASR grammars. However, we have also shown that we would probably get a much better performance if we took into account dialogue context in the semantic decoding process just as was done for the task of predicting the next dialogue move in order to choose the most appropriate SLM and for the task of re-ranking N-Best lists. This is something that would be easily done in an ISU-based framework by using the information state as an additional knowledge source when parsing the user input.

Chapter 3

Training statistical semantic parsers from a grammar generated corpus

In this chapter we describe models for semantic parsing that are designed to cover more complex semantic hierarchies. We describe the novel concept of the n-gram based semantic parser and compare it with the Hidden Vector State Model. As in the previous chapter these models are trained from corpora generated by a GF-task grammar, saving time and costs of manual annotation of training data. The domain is a tourist information task, where the driver of a car can ask for information about hotels, bars and restaurants in an invented town. This task was realised in English.

3.1 Training and Test Data

3.1.1 Collection of "Example" Utterances

A small corpus was collected for training and testing. We asked 9 co-researchers who were familiar with the task to submit a set of 10 "example" interactions with the system and a number of more advanced dialogues (see table 3.1). The data was divided into a development test set

Table 3.1: *"Example" interactions of training and test set.*

	Dev	Test
prompt sets	5	4
male / female users	10 / 6	7 / 2
native / non-native	12 / 4	5 / 4
sentences	353	311
Words	1605	1689

(dev) and a evaluation test set (test). It was taken care that the sets did not overlap. The dev set was mostly used as held out data for interpolation, selection of the best model, inspiration for

grammar design and other purposes. The test set was used for all the test runs done in the tourist information domain.

3.1.2 The tourist information domain

In the tourist information domain all slot value pairs and abstract concepts are linked to querying a database containing information about hotels, bars, restaurants and tourist attractions.

As abstract concepts we used what can be considered as the most common dialogue acts such as greetings (*hello, bye*), positive and negative answers to questions (*affirm, negate*), requests to repeat a system response (*repeat*), requests for an alternative entity (*reqalt*) or request for more information (*request_more*).

Near relations (*near*) make it possible to ask for something that is close to another place. And not relations allow recognition of phrases like "*not so expensive*"

Users can ask for general information about a bar, hotel or restaurant that fits their specifications (eg. *cheap, central*). This is reflected in the concepts *request_bar, request_hotel* and *request_restaurant*. In addition to that they can ask for the name, the telephone number, the price or the location of a place, mirrored as *request_name, request_phone, request_price* and *request_location*.

There is in general a slot for each field of the data base. The field *addr* contains street names, the field name was split into *hotelnames, barnames* and *restaurantnames*, the near relation can take tourist attractions, *attraction*, or names, the price range, *pricerange*, can take three values: cheap, moderate and expensive. The slot *pricelimit* reads a price that is specified as a number. The slot *food* takes different types of cuisines and *drinks* takes different beverages. There are five different town *areas*. Other slots take room types, stars, music, booking day, currency, numbers, ...

All slots accept the value "*dontcare*", indicating user responses like *the price is not important* or *doesn't matter where it is*. Even a unspecified "*dontcare*" is allowed, when the user just utters *I don't care* without indicating a specific slot. In this work the view on semantic parsing is that the parser can only use information provided in the sentence and does not take previous system responses into account. It is assumed that the dialogue manager will resolve any ambiguities that result from this.

The grammar was designed to cover user utterances that could occur in the pursuit of these tasks. The dev set was used as an inspiration for grammar writing. Especially for partial tasks where no example utterances could be found in the dev set other sources would have been used as well. The grammar writer knew the test set but did not consciously include phrases from it in the grammar.

3.1.3 Generating a Training Set with a GF-Grammar

The HTK task grammar that was used in D1.3 was ported to GF. The abstract syntax was used to model the words of a sentence. The concrete syntax was used to model the hierarchy of semantic

concepts. By removing the semantic concepts out of the concrete syntax a word grammar is generated as it was used for the training of statistical language models in D1.3. This has two advantages:

1. Removing all semantic concepts from the concrete syntax is very easy and can be done automatically. Therefore only one grammar must be maintained for training semantic parser and statistical language model of the speech recogniser.

2. The translate-function of GF can be used to implement a basic semantic parser. GF can translate from a word only grammar, using a concrete syntax with no semantic concepts, into a semantic grammar, which has semantic concepts specified in the concrete syntax. Both grammars use the same abstract syntax.

The corpus was generated using tools of GF and HTK. Both tools feature random generation of sentences. GF can output all different sentences that are possible in a grammar. The output combines the actual words that were uttered in a sentence with a semantic annotation. In this format the example sentence used above would have the following form:

I am looking for an up market French restaurant and a budget hotel in the town centre.

I am looking for request_restaurant(*an* pricerange(=expensive(*up market*))
food(=French(*French*)) *restaurant*) *and*
request_restaurant(*a* pricerange(=cheap(*budget*)) *hotel in the* area(=centre(*town centre*)))

This format is then translated into a HVS style input format or the input format for the n-gram based semantic parser.

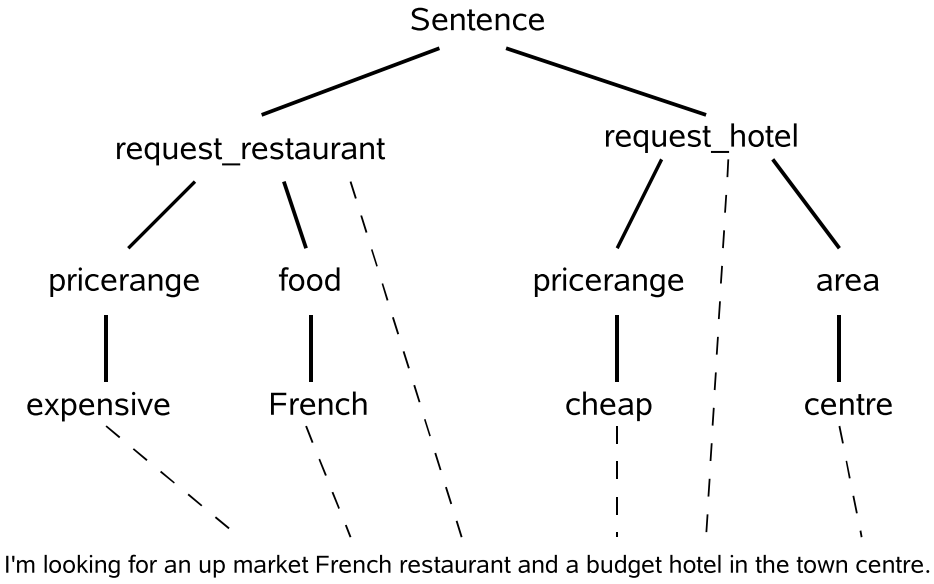
3.2 A hierarchical Model for Semantics

All semantic parsers described in this chapter are based on a semantic models that uses a tree hierarchy, where concepts that belong together can be grouped together under the same hierarchy level. Figure 3.1 illustrates this using an example sentence. The semantic hierarchy can either be written in a bracketed formalism or be displayed as a tree. It consists of slot-value pairs and abstract concepts. In the example sentence "request_restaurant()" or "request_hotel()" would be instances of abstract concepts. Their child nodes are usually other concepts. "pricerange = *expensive*" or "area = *centre*" would be instances of slot-value pairs.

Values are clearly linked to trigger words or phrases in the sentence. The relations between the word "*centre*" and the value *centre* or the words "*up market*" and the value *expensive* are indicated by dashed lines in the tree. Slots like "pricerange" or "food" are mostly determined by the values that they can take and whether the abstract concept above them can take them as a child node.

Abstract concepts are mostly determined by the parent node they can attach to and the child nodes they can expand into. They are also associated with words in the sentence, although this relation is not as explicit as in the case of values, as indicated by dashed lines in the tree.

Figure 3.1: *Example of a semantic hierarchy displayed as a tree or in a bracketed formalism.*



$$\begin{array}{ll} \text{Sentence}(& \text{request\_restaurant}(\text{pricerange} = \textit{expensive}, \quad \text{food} = \textit{French}) \\ & \text{request\_hotel}(\text{pricerange} = \textit{cheap}, \quad \text{area} = \textit{centre}) \\ ) & \end{array}$$

In the following semantic parsing will be viewed as a procedure to associate the correct semantic tree with a sentence or to associate only partial trees to the sentence.

3.3 Semantic parsing with a GF grammar

Many dialogue systems still use recognition grammars for semantic decoding. Since we have a GF grammar available, we can directly use it as a semantic parser.

The advantage of directly using a grammar is, that if a sentence gets parsed, the semantic tree or in the case of ambiguity the set of semantic trees will be perfect parses. The problem with this approach is that only trees that are explicitly coded in the grammar can be parsed. For a grammar that is used to generate training sentences for statistical models it is sufficient when phrases are roughly generated in the correct context. It is not necessary to model every detail, as it would be the case for a grammar based semantic parser.

Both dev set and test set were parsed with the generation grammar. 97 out of 353 sentences of the dev set could be parsed. Parsing of the test set succeeded in 72 out of 311 sentences. Especially long sentences providing a number of slot value pairs failed. For short answers the grammar worked quite well.

3.4 A n-gram Based Semantic Parser

A n-gram based semantic parser looks like an ordinary n-gram language model, the only difference is that the words are semantically enriched. The n-gram part of the model can be regarded as a framework to make sure the calculations can be done efficiently and from left to right, as the word stream comes out of the recogniser. The semantic model is coded in the training data.

3.4.1 Semantic Model

The semantic model of the n-gram based parser is encoded in the training data which consists of training sentences of semantically enriched words. These were generated with a grammar, as described in section 3.1.3. This material was converted into a stack based annotation. As displayed in figure 3.2 each semantically enriched word consists of the orthographic form and a semantic stack.

In the actual format used by the tools in this investigation different stack layers would be delimited by a dot ".". Values start with a ^ symbol and the last dot separated segment would always contain the orthographic form of the word. An example is given in figure 3.3.

Each orthographic word would have several entries in the vocabulary with different semantic stacks attached. The training of this model is simple, once a text is semantically annotated, since it only involves counting n-grams of semantically enriched words. Well established language modelling techniques such as backoff and smoothing can be used.

Figure 3.2: Stack based annotation for the training of n-gram based semantic parsers.

	request_restaurant		request_restaurant	request_restaurant		request_restaurant
			pricerange=*expensive*	pricerange=*expensive*		food=*French*
I am looking for	*an*		*up*	*market*		*French*

			request_hotel		request_hotel	request_hotel	
request_restaurant		request_hotel	pricerange=*cheap*	request_hotel	area=*centre*	area=*centre*	
restaurant	*and*	*a*	*budget*	*hotel in the*	*town*	*centre*	

Figure 3.3: Actual format used for the training of n-gram based semantic parsers.

```
I
AM
LOOKING
FOR
REQUEST_RESTAURANT.AN
REQUEST_RESTAURANT.PRICERANGE.^EXPENSIVE.UP
REQUEST_RESTAURANT.PRICERANGE.^EXPENSIVE.MARKET
REQUEST_RESTAURANT.FOOD.^FRENCH.FRENCH
REQUEST_RESTAURANT.RESTAURANT
AND
REQUEST_HOTEL.A
REQUEST_HOTEL.PRICERANGE.^CHEAP.BUDGET
REQUEST_HOTEL.HOTEL
REQUEST_HOTEL.IN
REQUEST_HOTEL.THE
REQUEST_HOTEL.AREA.^CENTRE.TOWN
REQUEST_HOTEL.AREA.^CENTRE.CENTRE
```

3.4.2 n-gram semantic decoding

Parsing a sentence involves expanding all orthographic words of a sentence into a network with all words given all possible semantic prefixes and searching the best path in this network. The Viterbi Algorithm provides an efficient solution to this problem. In this work, the model would first consider only transitions of the order of bigrams or higher. Only if no bigram transition was found it would resort to using unigrams.

Apart from the n-gram or Markov Assumption this model does not presuppose any semantic theory. The semantic theory is coded in the semantic annotation of the training material. Using a semantic stack is only one possibility of realising n-gram based semantic parsers.

3.4.3 Experiments and Results

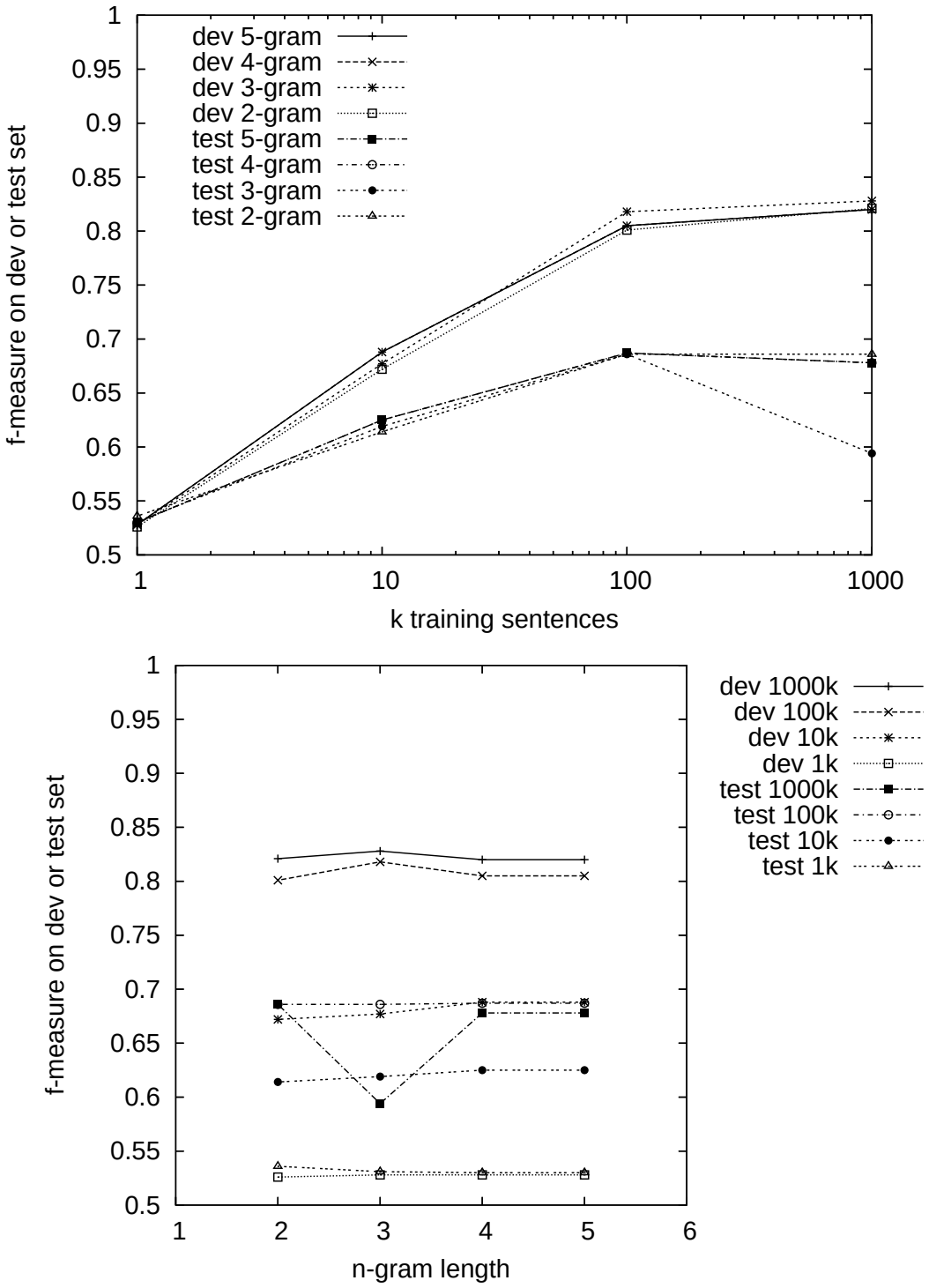
To investigate how much training data a n-gram based semantic parser needs to work properly and to investigate the influence of the length of the word history *n* on the quality of the parsing result, the grammar of section 3.1.3 was used to randomly generate a training corpus of one million sentences of semantic words. The n-gram based semantic parser was trained on the first 1000, 10000, 100000 and 1000000 words of the corpus. The length of the word history *n* was varied between 2 and 5. All n-grams were built with modified Kneser-Ney backoff. The evaluation was carried out on two test sets consisting of invented dialogues. The dev set was used to build the grammar, which means the grammar would contain a lot of the phrases that are in this data set. It is very likely that some of those appeared in the training data for the n-gram based semantic parser. Similarly, the test set is only based on invented dialogues and this is not ideal, but it is currently the best available approximation to an independent test set until recordings of interactions of real users with a real system become available.

The parses of different models were evaluated against manual reference annotations. Each model was evaluated twice:

- In the deep evaluation, a tree was assumed as the theoretical model. The evaluation was performed top to bottom, which means, that the full stack must be identical with the reference to be counted as correct.
- In the shallow evaluation, only leave nodes were evaluated. Hierarchy was only considered in a limited number of concepts e.g. in “not” relations as in not.pricerange=*expensive* relating to the phrase *not too expensive* or in near relations, like near.hotelname=”*Royal Hotel*” corresponding to the phrase *near the Royal Hotel*. Leaf nodes can be concept-value pairs or just concepts. In the case of short and simple sentences this kind of evaluation is often sufficient and realistic.

In both cases identical items in the parse were merged and only evaluated once. That means if request_bar was attached to the words *a* and *bar*, this would be regarded only as one instance of this concept. Precision and recall were calculated. The harmonic mean of these two values, also called f-measure was used as an evaluation metric.

Figure 3.4: *Results of a deep evaluation of the n-gram parser.*



The top plot of figure 3.4 shows two distinct clusters of lines. It is not surprising that all models performed better on the dev set than on the test set and that increasing the number of training sentences gives better results. Above 100000 sentences the performance on the dev set improve only marginal. On the test set the results even decrease. All models achieve their best performance of 69% on the test set for a training set of 100000 sentences. Results for 4-grams and 5-grams are identical, probably because non of the 5-grams in the models occurs in the test set or dev set. The best performance of the dev set was 83% and achieved with a trigram model trained on one million words. This is the same model that produces the outlier on the test set. The bottom plot of figure 3.4 shows that bigrams and trigrams yield almost identical results as higher order models. It seems that the semantic annotation of the previous word carries a lot of information already such that it is not necessary to look further ahead.

Figure 3.5: *Results of a shallow evaluation of the n-gram parser.*

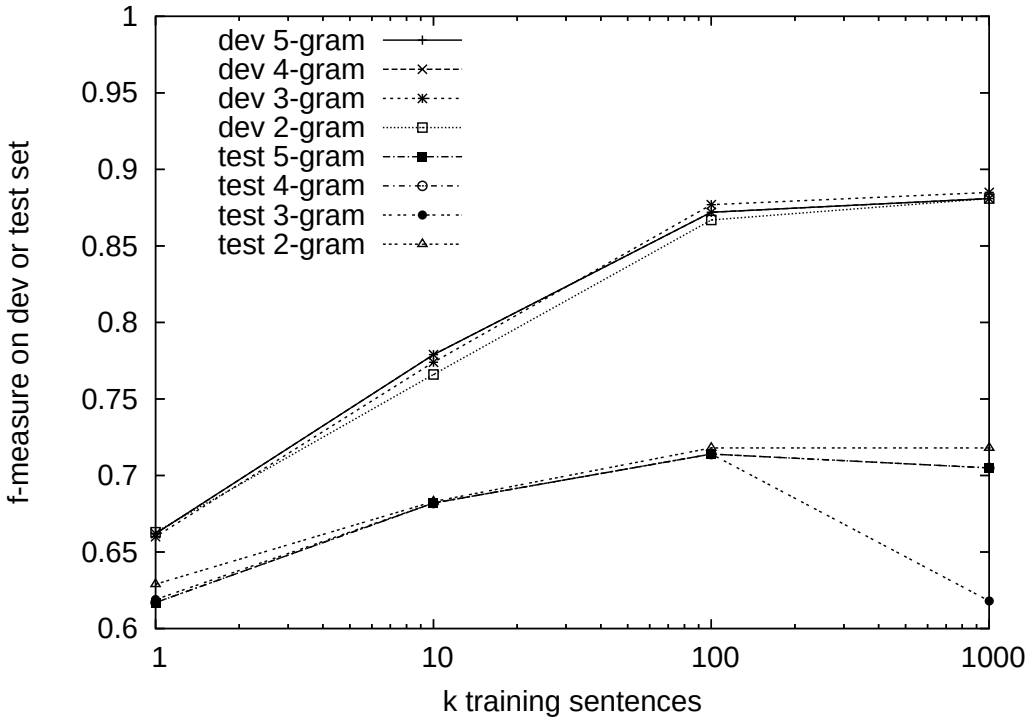


Figure 3.5 shows the results of the shallow evaluation. The general impression is very similar to the deep evaluation, however the f-measures of the dev set is around 5% to 10% higher than in the deep evaluation. The best result on the dev set was the one million word trigram with 89% and for the test-set it was the 100000 and the one million word bigram models that reached the best performed best with 72%.

Bigrams and trigrams seem to be very well suited for n-gram semantic parsing. This is an important result, as it allows the parsing algorithm to be kept simple and efficient.

3.5 A Hidden Vector State Semantic Parser

The explanation of the Hidden Vector State Model in this section follows in large parts the description given in [HY05]. Considering the semantic parse tree shown in figure 3.1, the semantic information relating to each word is completely described by the sequence of semantic concept labels extending from the preterminal node to the root node. If these semantic concept labels are stored as a single vector, then the parse tree can be transformed into a sequence of vector states similar to the one shown in section 3.4.1. Viewing each vector state as a hidden variable, the whole parse tree can be converted into a first order vector state Markov model.

Each vector state is in fact equivalent to a snapshot of the stack in a push-down automaton. Indeed, given some maximum depth of the parse tree, any Probabilistic Context-Free Grammar (PCFG) formalism can be converted to a first-order vector state Markov model. If we view each vector state as a stack, then state transitions may be factored into a stack shift by n positions followed by a push of one or more new preterminal semantic concepts relating to the next input word. If such operations are unrestricted, then the state space will grow exponentially and the same computational tractability issues of hierarchical HMMs are incurred. However, by imposing constraints on the stack operations, the state space can be reduced to a manageable size. Possible constraints to introduce are limiting the maximum stack depth and only allowing one new preterminal semantic concept to be pushed onto the stack for each new input word. These constraints ensure that the size of the underlying probability tables are linear in stack depth, number of concept labels, and vocabulary size. Such constraints effectively limit the class of supported languages to be right-branching. Although left-branching structures do exist in English, the majority of sentences can be represented as right-branching structures. In addition, right-branching structures are generally preferred because they reduce the working memory needed to represent a sentence [Phi95].

Although any parse tree can be converted into a sequence of vector states, it is not a one-to-one mapping and ambiguities may arise. However for the semantic parsing task defined here, what we are interested in is not the exact parse tree to be recovered, but are the concept/value pairs to be extracted. Even if there are ambiguities, the extracted concept/value pair will be the same. For example, assuming A and B are semantic concept labels and x and y are words, the partial parse trees B(A(x) A(y)) and B(A(x y)) would share a common HVS representation and hence would yield the same concept/values B.A=x and B.A=y. If the entities x and y were actually distinct, then the preterminal labels would have to be made unique.

3.5.1 Definition of the HVS model

The joint probability $P(N,C,W)$ of a series of stack shift operations N, a concept vector sequence C, and a word sequence W can be decomposed as follows

$$P(N,C,W) = \prod_{t=1}^{T} P(n_t|W_1^{t-1},C_1^{t-1})P(c_t[1]|W_1^{t-1},C_1^{t-1},n_t)P(w_t|W_1^{t-1},C_1^t) \qquad (3.1)$$

where

- C_1^t denotes a sequence of vector states $c_1..c_t$. c_t at word position t is a vector of D_t semantic concept labels (tags), i.e. $c_t = [c_t[1], c_t[2], .., c_t[D_t]]$ where $c_t[1]$ is the preterminal concept immediately dominating the word w_t and $c_t[D_t]$ is the root concept,
- $W_1^{t-1}C_1^{t-1}$ denotes the previous semantic parse up to position $t-1$,
- n_t is the vector stack shift operation and takes values in the range $0, .., D_{t-1}$,
- $c_t[1] = c_{w_t}$ is the new preterminal semantic tag assigned to word w_t at word position t.

The stack transition from $t-1$ to t given preterminal semantic concept tag c_{w_t} for word w_t is

$$c_t[1] = c_{w_t} \quad (3.2)$$
$$c_t[2..D_t] = c_{t-1}[(n_t+1)..D_{t-1}] \quad (3.3)$$
$$D_t = D_{t-1} + 1 - n_t \quad (3.4)$$

Thus n_t defines the number of semantic tags which will be popped off the stack before pushing on c_{w_t}. The case $n_t = 0$ corresponds to growing the stack by one element i.e. entering a new semantic tag. The case $n_t = 1$ corresponds to simply replacing the preterminal at word position $t-1$ by c_{w_t} at word position t, the rest of the stack being unchanged. The case $n_t > 1$ corresponds to shifting the stack i.e. popping off one or more semantic tags.
Equation 3.1 is approximated by

$$P(n_t|W_1^{t-1}, C_1^{t-1}) \approx P(n_t|c_{t-1}) \quad (3.5)$$
$$P(c_t[1]|W_1^{t-1}, C_1^{t-1}, n_t) \approx P(c_t[1]|c_t[2..D_t]) \quad (3.6)$$
$$P(w_t|W_1^{t-1}, C_1^t) \approx P(w_t|c_t) \quad (3.7)$$

3.5.2 Training assumptions

The HVS model needs the following resources for training:

- *A set of domain specific lexical classes*. For example, in an tourist information domain, it is possible to group all names of tourist attractions into one single class `ATTRACTION`. Such domain specific classes can normally be extracted automatically from the application domain database schema.
- *Abstract semantic annotation for each utterance*. Such an annotation need only list a set of valid semantic concepts and the dominance relationships between them without considering the actual realised concept sequence or attempting to identify explicit word/concept pairs.

The provision of abstract annotations implies that the dialogue designer must define the semantics that are encoded in each training utterance but need not provide an utterance level parse. It effectively defines the required input-output mapping whilst avoiding the need for expensive tree-bank style annotations. For example, in a tourist information domain, a dialogue system designer may define the following hierarchical semantic relationships:

- REQUEST_RESTAURANT(PRICERANGE NEAR(ATTRACTION))
- REQUEST_PRICE(RESTAURANTNAME)
- ...

Having defined such a set of hierarchical semantic relationships, annotation is simply a method of associating the appropriate semantics with each training utterance and does not require any linguistic skills. For example, when building a system from scratch, a dialogue designer can take each possible abstract schema in turn and give examples of corresponding natural language forms, as in

REQUEST_RESTAURANT(PRICERANGE(X) NEAR(ATTRACTION(D)))→

1. I would like a X restaurant near the D.
2. find me a restaurant close to D that is X.
3. I am looking for something X to eat somewhere around D.

In this experiment we used a grammar to generate a corpus that provides both the surface form and the corresponding semantic annotation.

3.5.3 Training

The first step needed to train the HVS model is to replace all class members by their corresponding class names. Where there is ambiguity, the class covering the largest span of words is replaced first. Where a word or phrase may occur in more than one class, the first class encountered is chosen arbitrarily.

In a second step the vector state sequence will be expanded from the abstract annotation and the appropriate concepts will be attached to the corresponding lexical Concepts. If there are more words than vector states in the sequence, new vector states with the concept DUMMY in the bottom position will be added.

Note that this final set of vector states only provides the set of valid semantic vector states that can appear in the parse results of the current utterance. As explained further below, this set is used as a constraint in the EM re-estimation algorithm. It does not define the actual vector state transition sequence. Further note that the total number of distinct vector states required to use the HVS model for a particular application can be enumerated directly from this expanded vector state list.

The system only allows the DUMMY tag to appear in preterminal positions, therefore, for consecutive irrelevant word inputs, the model will stay in the same vector state. Only when a relevant word input is observed will the DUMMY tag together with zero or more preceding semantic tags be popped off from the previous vector state stack and a new preterminal tag will be pushed into the stack accordingly.

The training starts with a flat initialisation. Then all parameters are iteratively refined using an EM based re-estimation procedure. There are no explicit word level annotations in the training corpora, hence parameter estimation based on event counts cannot be used and forward-backward estimation must be applied instead. Let the complete set of model parameters be denoted by λ.

EM-based parameter estimation aims to maximise the expectation of $L(\lambda) = \log P(N,C,W|\lambda)$ given the observed data and current estimates.
During training, two constraints are applied in the estimation process:

1. For each utterance, a state transition is only allowed if both incoming and outgoing states can be found in the corresponding semantic annotation.

2. If the observed word is a class name (such as ATTRACTION), then only semantic concepts (states) which contain this class name can be associated with the word (eg NEAR.ATTRACTION, but not REQUEST_BAR). In addition, in order to cater for irrelevant words, the DUMMY tag is allowed everywhere. That is, state transitions from or to the DUMMY state are always allowed.

The following example illustrates how these two constraints are applied. Consider again the annotation for the utterance "I would like a cheap restaurant near the Castle" which was in abstract form

```
REQUEST_RESTAURANT  REQUEST_RESTAURANT+DUMMY
REQUEST_RESTAURANT+PRICERANGE(X)  REQUEST_RESTAURANT+PRICERANGE+DUMMY
REQUEST_RESTAURANT+NEAR  REQUEST_RESTAURANT+NEAR+DUMMY
REQUEST_RESTAURANT+NEAR+ATTRACTION(D)  REQUEST_RESTAURANT+NEAR+ATTRACTION(D)+DUMMY
```

The transition from REQUEST_RESTAURANT+PRICERANGE(X) to REQUEST_RESTAURANT is allowed since both states can be found in the semantic annotation. However, the transition from REQUEST_RESTAURANT to REQUEST_BAR is not allowed as REQUEST_BAR is not listed in the semantic annotation. Also, for the lexical item X in the training utterance, the only valid vector state is REQUEST_RESTAURANT+PRICERANGE(X) since X has to be bound with the preterminal tag PRICERANGE.

3.5.4 Experiments and Results

The same grammar generated corpus was used as in the experiments described in section 3.4.3. All other resources necessary for the training of a HVS model such as

- training sentences,
- abstract semantic annotation and
- lexical classes with their corresponding word string mappings

were generated by the grammar as well. The HVS stack size was set to 5 and Witten-Bell smoothing was used. As in the previous experiment a shallow and a deep evaluation was carried out. Concept value pairs were only evaluated if the terminal concept and the lexical class would match or if the terminal concept was DUMMY and the pre-terminal concept matched the lexical class.

Figure 3.6: *Results of a shallow evaluation of the Hidden Vector State parser.*

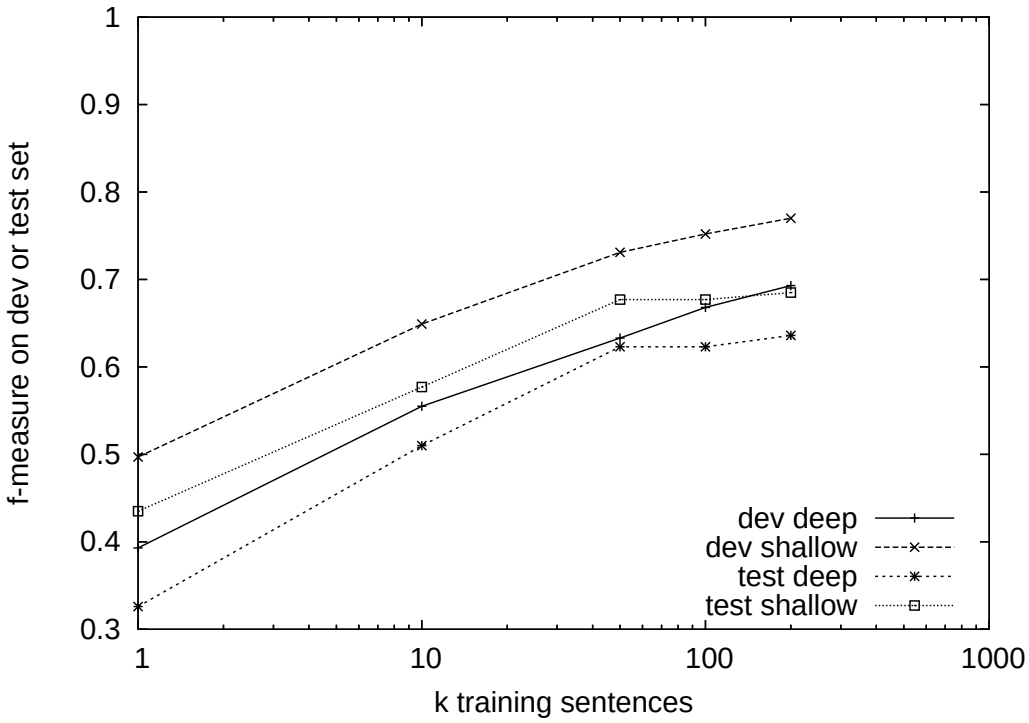


Figure 3.6 shows that the performance of the HVS model improves as the size of the training set increases. For all graphs the best f-measures are obtained at 200000 training sentences. The dev set reaches 69% in the deep evaluation. This improves to 77% in the shallow evaluation. The best f-measure on the test set is 63% in the deep evaluation and 69% in the shallow evaluation.

3.6 Conclusion

Both statistical models show flexibility in covering unseen sentences. The n-gram parser gives better results than the HVS model. Training a n-gram parser involves only counting semantically enriched words as provided by the grammar. This is a much simpler task than aligning a semantic annotation using EM in the case of the HVS model. Both n-gram parser and HVS model require careful design of the concept hierarchies to work well. Although the general impression is, that the HVS model in its current implementation needs more tweaking to get it to work. Especially the implementation of a "DONTCARE" value, which is a value that is possible for all slots and can even be unspecified[1].

The results in this investigation are a lower than those obtained by work on the ATIS corpus, which is the standard corpus used for similar investigations. There are two possible explanations

[1] In this case it can only be classified by using the previous system dialogue act as context.

for that. The first is that in this investigation the models were trained on randomly generated corpora and not on a proper training set. The second is that details in the task definition such as the integration of a DONTCARE value may have caused our task to be more difficult than ATIS.

Both statistical models are fit for being used in a dialogue system. Bootstrapping with a grammar generated corpus works for both models, although it is necessary to further investigate how both models can be improved by training them on real user data, without going through the pain of manually annotating large data sets. In this respect the HVS model seems to be better suited.

Both statistical models outperformed the GF grammar by some margin. In real recognition experiments the grammar would have probably scored better, forcing the speech recogniser on a semi-optimal path through the grammar instead of failing. This is a bit of an unfair comparison, as the grammar was not designed to explicitly cover all sentences, but to generate a corpus for the training of statistical models.

Future work will involve the integration of both decoders in a speech dialogue system and evaluating their performance under real world conditions.

Chapter 4

Conclusion and Future Work

Semantic decoders in the form of dialogue act taggers or hierarchical parsers are an essential component of a spoken dialogue system. When SLM-based ASR is used it can be particularly problematic since the recognised word sequence may contain speech recognition errors or may not be grammatical. That means robust parsing is necessary. Statistical semantic decoders provide a solution to this problem, but they need training data. Following from D1.3, this work has investigated the effectiveness of using small development grammars to generate bootstrap training corpora.

Investigations were conducted in Swedish in an MP3 domain using pattern matching techniques such as TiMBL and MBT. Although these taggers were not capable of capturing deep semantic relationships they were sufficient for the semantics of the domain. Both methods worked well and yielded 78% accuracy. This means an important boost in performance in comparison to the more restricted parsing behaviour of the GF grammar. Although the semantic decoders obtained have not yet been integrated in the dialogue system they have been used successfully for other decoding tasks.

For future work we will consider to make GF itself more robust. One possibility is to develop an agenda-driven chart-based parsing algorithm for GF which will derive a set of parse items even if the recognition is unsuccessful. By using a number of deduction and induction rules one could paste these items, representing substrings of the input, into bigger items and thus end up with an item covering the entire input. Using induction rules will guarantee a result but also generate a cost representing the difference between the input being in the language of the grammar or not.

Another strategy that we are considering is what we call ”Shake-and-Bake parsing” based on the notion of Shake-and-Bake semantics [KS85]. The concept is quite easy. When it is not possible to recognise an input string with a concrete grammar you check if the fully instantiated items in the obtained parse chart can be meaningful according to the abstract grammar. This means that when given an input such as ”runs John” we would have the abstract rule `S := NP VP` (meaning that we have an S if there is an NP and a VP without considering the order of the two latter); the concrete rules `S -> NP VP`, `NP -> John` and `VP -> runs` and the fully instantiated items for ”John” as an NP and ”runs” as an VP. These items cannot be joined to an S by the concrete rule but they fulfill the requirements for deriving an S by the abstract rule. We can therefore use the

obtained abstract syntax of an S with its NP mapped to ”John” and its VP mapped to ”runs” to linearise a semantic representation.

A second series of experiments was carried out on an English tourist information domain using hierarchical semantic decoders such as the Hidden Vector State Model an n-gram based parser. Both models are capable of capturing semantic hierarchies of sentences. Evaluation of the HVS model resulted in f-measures of 63% for the deep evaluation and 69% for the shallow evaluation. The n-gram based parser reached an f-measure of 69% for the deep evaluation and 72% in the shallow evaluation. Comparision of our results with results that we achieved on the ATIS corpus using the HVS model suggests that our tourist information task is probably a bit harder and that there should be room for improvement when using real training data instead of a grammar generated corpus. We are planning to include robust semantic decoders in a speech dialogue system to evaluate their performance in the presence of real users.

Both investigations showed independently that statistical semantic parsers have a clear advantage over rule based methods. Similar to our results in deliverable D1.3 we could show that training semantic parsers on corpora that were generated by hand crafted grammars is a good method for bootstrapping dialogue systems. Parsers created in this way can also be used to provide a raw annotation for training data, minimising the effort of manual corrections.

Bibliography

[Car96] Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22, 1996.

[DMAM94] J. Dowding, R. Moore, F. Andry, and D. Moran. Interleaving syntax and semantics in an efficient bottom-up parser. In *Proc. of the 32nd Annual Meeting of the Association for Computational Linguisics*, 1994.

[DZvdBvdS03] Walter Daelemans, Jakub Zavrel, Antal van den Bosch, and Ko van der Sloot. MBT: memory based tagger, version 2.0, Reference Guide., 2003.

[DZvdSvdB01] Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. TiMBL: tilburg memory-based learner - version 4.0 Reference Guide, 2001.

[EAB+06] Stina Ericsson, Gabriel Amores, Björn Bringert, Robin Cooper, David Hjelm, Rebecca Jonson, Staffan Larsson, Pilar Manchon, David Milward, and Guillermo Perez. Software illustrating a unified approach to multimodality and multilinguality in the in-home domain. Status Report T1.6, TALK project, 2006.

[FST98] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 1998.

[HY05] Yulan He and Steve Young. Semantic processing using the hidden vector state model. *Computer Speech and Language*, 19(1):85–106, 2005.

[Jon06] Rebecca Jonson. Generating statistical language models from interpretation grammars in dialogue systems. In *Proceedings of 11th Conference of the European Association of Computational Linguistics*, pages 57–65, Trento, Italy, 2006.

[KS85] E Klein and I. Sag. Type-driven translation. *Linguistics and Philosophy*, 8:163–201, 1985.

[Lar02] Staffan Larsson. *Issue-based Dialogue Management*. PhD thesis, Göteborg University, 2002.

[LBC+05] Peter Ljunglöf, Björn Bringert, Robin Cooper, Ann-Charlotte Forslund, David Hjelm, Rebecca Jonsson, Staffan Larsson, and Aarne Ranta. The TALK grammar library: an integration of GF with TrindiKit. Deliverable D1.1, TALK Project, 2005.

[LdBKC04] P. Lendvai, A. Van den Bosch, E. Krahmer, and S. Canisius. Memory-based robust interpretation of recognised speech. In *Proceedings of SPECOM '04, 9th International Conference "Speech and Computer"*, pages 415–422, St. Petersburg, Russia, 2004.

[MAB+05] David Milward, Gabriel Amores, Tilman Becker, Nate Blaylock, Malte Gabsdil, Staffan Larsson, Oliver Lemon, Pilar Manchon, Guillermo Perez, and Jan Schel. Integration of ontologies and the ISU approach for multimodal dialogue: illustration for in-car information. Deliverable D2.1, TALK project, 2005.

[PB96] Cosmin Popovici and Paolo Baggia. Specialized language models using dialogue predictions. *CoRR*, cmp-lg/9612002, 1996.

[Phi95] C. Phillips. Right association in parsing and grammar. In C. Schütze, K. Broihier, and J. Ganger, editors, *Papers on Language Processing and Acquisition*, pages 37–93. 1995. MITWPL 26.

[PKIW98] P.Taylor, S. King, S. Isard, and H. Wright. Intonation and dialogue context as constraints for speech recognition. 1998.

[PM98] M. Poesio and A. Mikheev. The predictive power of game structure in dialogue act recognition: Experimental results using maximum entropy estimation. In *ICSLP'98*, 1998.

[PTG+92] R. Pieraccini, E. Tzoukermann, Z. Gorelow, E. Levin, and J. Gauvain. Progress report on the CHRONUS system: ATIS benchmark results. In *Proc. of the DARPA Speech and Natural Language Workshop*, 1992.

[Ran04] A. Ranta. Grammatical framework: A type-theoretical grammar formalism. *Journal of Functional Programming*, 14(2):145–189, 2004.

[REKK96] N. Reithinger, R. Engel, M. Kipp, and M. Klesen. Predicting dialogue acts for a speech-to-speech translation system. In *Proc. ICSLP '96*, volume 2, pages 654–657, Philadelphia, PA, 1996.

[SCB+00] Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373, 2000.

[SCVS98] Ken Samuel, Sandra Carberry, and K. Vijay-Shanker. Dialogue act tagging with transformation-based learning. In *Proceedings of COLING-ACL*, pages 1150–1156, 1998.

[Sen92] Stephanie Seneff. Robust parsing for spoken language systems. In *Proceedings of the ICCASP*, 1992.

[SMSM97] R. Schwartz, S. Miller, D. Stallard, and J. Makhoul. Hidden understanding models for statistical sentence understanding. In *Proc. of the ICASSP*, 1997.

[WI96] W. Ward and S. Issar. Recent improvements in the CMU spoken language understanding system. In *Proc. of the ARPA Human Language Technology Workshop*, 1996.

[WJRY06] Karl Weilhammer, Rebecca Jonson, Aarne Ranta, and Steve Young. SLM generation in the grammatical framework. Deliverable D1.3, TALK Project, 2006.

[WPI99] H. Wright, M. Poesio, and S. Isard. Using high level dialogue information for dialogue act recognition using prosodic features. In *In DIAPRO-1999*, 1999.

[You04] Steve Young. *ATK: An Application Toolkit for HTK, Version 1.4.1*. Machine Intelligence Laboratory, Cambridge University Engineering Dept, Trumpington Street, Cambridge, CB2 1PZ, July 2004. http://htk.eng.cam.ac.uk/develop/atk.shtml.

[YSW05] Steve Young, Matt Stuttle, and Karl Weilhammer. Integrated SLMs for multiple input modes. Status Report T1.4s2, TALK project, 2005.

[YY06] Hui Ye and Steve Young. A clustering approach to semantic decoding. In *Proc. of the ICSLP*, Pittsburgh, 2006.